Sommaire	Numéro de page

Liste des figures

Introduction Générale :

L'évolution des langages de programmation a amené de nouveaux outils aidant à la conception des applications en informatique. En effet, l'événement de l'orienté objet facilite l'abstraction du problème à résoudre en fonction des données du problème lui-même (par l'utilisation des classes et d'objets).

C'est ainsi, que la réalisation de logiciels de gestion demeure une activité professionnelle difficile. Malgré les progrès apportés par le génie logiciel, les développements d'application répondant aux besoins exprimés se rationalisent lentement. En revanche, l'offre d'outils de développement ne cesse de croitre et l'importance des langages de programmation est toujours prépondérante. De plus en plus d'applications utilisent même plusieurs langages de programmation dans le cadre d'un projet unique. Nous constatons enfin, un intérêt grandissant pour issus des technologies objets.

L'objectif de notre PFE est faire la gestion d'une banque en se basant sur une architecture de type client/serveur et en utilisant le mécanisme JAVA RMI.

Notre rapport est organisé comme suit :

Le premier chapitre introduit la gestion d'une banque en général, présente la programmation orientée objet qu'on a employé pour réaliser l'application et préciser le mécanisme utilisé RMI (Remote Method Invocation) et enfin parler des outils avec lesquels on a réalisé l'application comme Java et NetBeans, ensuite nous allons présenter MySQL (en utilisant EasyPHP) qui est un serveur de base de données relationnelle, et enfin nous allons présenter l'architecture client-serveur utilisée dans le cadre de ce PFE.

Le deuxième chapitre est consacré à la présentation de l'application.

Chapitre I : Architecture et outils utilisés.

I.1 Introduction :

Dans cette partie, nous allons parler un petit peu de la gestion d'une banque et ses services et évoquer et définir tous les outils utilisés dans ce projet.
On va commencer par une présentation du langage **JAVA**, suivi par la présentation de **NetBeans** qui est un environnement de développement intégré (EDI), ensuite nous allons présenter **MySQL** qui est un serveur de base de données relationnelle, ensuite on va donner une petite présentation de l'architecture Client/serveur et enfin parler du mécanisme utilisé **Java RMI.**

I.2 Gestion d'une banque :

La banque doit pouvoir atteindre plusieurs objectifs dans le cadre de la gestion quotidienne des comptes bancaires. Ces objectifs sont bien sûr complémentaires. Il lui faut :

- Maîtriser les risques associés aux opérations bancaires réalisées sur le compte.
- Améliorer la qualité des services rendus aux clients.
- Rentabiliser ses comptes en équipant la clientèle des produits et services de la gamme.

Les services disponibles de notre banque sont :

1/ Services pour le Client :

- Opérations sur le comptes bancaire (consulter, créditer, débiter, supprimer).
- Virement d'un compte à un autre compte.
- La sécurité totale des comptes.

2/Administrateur :

Les administrateurs sont responsables de l'inscription des clients(d'ouvrir les comptes courants et épargnes) comme ils ont le même droit de profiter des services tel un client et ils ont de plus :

- Le droit de voir tout l'historique et toutes les opérations qui sont faites.

I.3 Les outils utilisés:

I.3.1 Java :

Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels qu'UNIX, Microsoft Windows, Mac OS ou Linux... C'est la plateforme qui garantit la portabilité des applications développées en Java.

Le langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épurée des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que l'héritage multiple remplacé par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.). [1]

I.3.1.1 Historique :

En raison des difficultés rencontrées avec C++, il était préférable de créer un nouveau langage autour d'une nouvelle plate-forme de développement. Deux développeurs de chez SUN, James Gosling et Patrick Naughton se sont attelés a cette tâche afin d'avoir une plate-forme et un langage idéal pour le développement d'applications sécurisées, distribuées, robuste et portable sur de nombreux périphériques et systèmes embarqués interconnectés en réseau mais également sur Internet (clients légers) et sur des stations de travail (clients lourds), n'oubliant pas la richesse de ses API fournis en standard ou par des tiers commerciaux ou libres.

D'abord surnommé C++-- (C++ sans ses défauts) puis OAK, mais il s'agissait d'un nom déjà utilisé dans le domaine informatique, il fut finalement baptisé Java mot d'argot voulons dire café en raison des quantités de café ingurgité par les programmeurs et notamment par ses concepteurs.

Java est puis rachetée par Oracle en 2010 qui a réussi à obtenir une très grande célébrité en seulement quelques années grâce à ces qualités.

Aujourd'hui Java est largement utilisée notamment en entreprises et pour les applications des appareils mobiles.

I.3.1.3 La programmation Orientée Objet :

Il est impossible de parler de Programmation Orientée Objet sans parler d'*objet*, bien entendu. Tâchons donc de donner une définition aussi complète que possible d'un *objet*.
Un objet est avant tout une structure de données (Objet =Données+Méthodes). Autrement, il s'agit d'une entité chargée de gérer des données, de les classer, et de les stocker sous une certaine forme. En cela, rien ne distingue un *objet* d'une quelconque autre structure de données. La principale différence vient du fait que l'*objet* regroupe les données et les moyens de traitement de ces données.

Un *objet* rassemble de fait deux éléments de la programmation procédurale.

- Les **champs** :

  Les *champs* sont à l'objet ce que les variables sont à un programme : ce sont eux qui ont en charge les données à gérer. Tout comme n'importe quelle autre variable, un *champ* peut posséder un type quelconque défini au préalable : nombre, caractère... ou même un type objet.

- Les **méthodes** :

  Les *méthodes* sont les éléments d'un objet qui servent d'interface entre les données et le programme. Sous ce nom obscur se cachent simplement des procédures ou fonctions destinées à traiter les données.

Les champs et les méthodes d'un objet sont ses **membres**.

Si nous résumons, un *objet* est donc un type servant à stocker des données dans des *champs* et à les gérer au travers des *méthodes*.
Si on se rapproche du Pascal, un objet n'est donc qu'une extension évoluée
des *enregistrements* (type **record**) disposant de procédures et fonctions pour gérer les champs qu'il contient.

Encapsulation des données :

L'accès aux données des objets est réglementé car les données privées sont accessibles uniquement par les fonctions membres. Les données publiques sont accessibles directement par l'instance de l'objet par conséquence un objet n'est vu que par ses spécifications. Une modification interne est sans effet pour le fonctionnement général du programme et une meilleure réutilisation de l'objet.

L'Héritage :

Permet de définir les bases d'un nouvel objet à partir d'un objet existant. Aussi le nouvel objet hérite des propriétés de l'ancêtre et peut recevoir de nouvelles fonctionnalités. Enfin la possibilité de la réutilisation de l'objet.

I.3.1.4 JDK (Java Development Kit):

L'écriture des applets et des applications Java nécessite l'utilisation d'outils de développement tels que le kit JDK. Ce dernier comprend l'environnement JRE, le compilateur Java et les API Java.

I.3.1.5 JVM (Machine Virtual Java):

La machine virtuelle java n'est qu'un aspect du logiciel Java, lié à l'interaction avec le web. La JVM est incluse dans le téléchargement du logiciel Java et permet l'exécution des applications Java.

La JVM est un des éléments les plus importants de la plate-forme Java : une bonne compréhension de son fonctionnement et de certains des concepts qu'elle met en œuvre est très importante pour obtenir les meilleures performances avec certaines applications.[2]

I.3.1.6 Les caractéristiques de Java :

- Java est interprétée.
- Java est portable.
- Java est orienté objet.
- Java est simple.
- Java est fortement typée.
- Java assure la gestion de la mémoire.
- Java est sûre.
- Java est économique.
- Java est multitâche. [3]

I.3.1.7 Java swing:

Swing est une boîte à outils de l'interface graphique principale pour le langage de programmation Java. Il s'agit d'une partie de la JFC (Java Foundation Classes), qui est une

API pour fournir une interface utilisateur graphique pour des programmes Java. Il est entièrement écrit en Java. [4]

I.3.2 ArgoUML:

Est un outil d'aide à la conception orientée objet.

- Une application multiplateforme :

  ArgoUML est entièrement codé en Java 1.2 et utilise les classes de base de Java (Java Foundation Classes). Ceci permet à ArgoUML de fonctionner sur pratiquement n'importe quelle plateforme munie d'une machine virtuelle Java.

- Standard UML

  ArgoUML est conforme avec la norme UML 1.3 définie par l'OMG. Le code pour la représentation interne d'un modèle UML est complètement produit suivant les spécifications de l'OMG. Pour se faire, une bibliothèque spéciale de metamodel (NSUML) a été développée par la société Novosofts sous licence GPL. Ceci rend ArgoUML extrêmement flexible pour s'ajuster aux nouvelles normes UML à venir. Cependant quelques caractéristiques avancées d'UML ne sont pas encore disponibles dans les diagrammes. Notamment, il n'existe pas encore de diagramme de séquence. [5]

I.3.3 L'IDE NetBeans :

I.3.3.1 Historique :

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'EDI NetBeans, jusqu'à ce qu'il soit acheté par Sun en 1999. Sun place le projet sous double licence CDDL et GPL v2 en juin de l'année suivante.

I.3.3.2 Présentation :

NetBeans IDE est un environnement de développement intégré (EDI) modulaire basé sur des normes, écrit dans le langage de programmation Java. Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java et en une plate-forme d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application. [6]

Voici une liste de choses que l'IDE fait pour simplifier le développement d'Applications Web:

- Fournit un serveur Web Tomcat pour déployer, tester et déboguer vos applications.
- Définit le fichier et la structure de dossier d'une application web pour vous.
- Génère et maintient le contenu des descripteurs de déploiement.
- S'assure que les fichiers de configuration qui apparaissent dans le dossier WEBINF de votre application ne sont pas effacés lorsque vous exécutez la commande Clean pour enlever les résultats des builds précédents.
- Fournit une coloration syntaxique, complétion de code (L'éditeur texte reconnait les mots clés du code source et les colorie selon leur utilisation et leur type : mot clé du langage, variable, fonction, chaine de caractère, commentaire...etc.).
- Fournit un support de débogage compréhensif, incluant le mode pas à pas dans les fichiers JSP et le traçage des requêtes HTTP.

I.3.4 EasyPHP et MySQL :

I.3.4.1 Présentation :

EasyPHP permet d'installer MySQL, une base de données, le troisième et inséparable membre du trio Apache/PHP/MySQL. Une base de données est un programme permettant de gérer une grande quantité de données en les organisant sous forme de tables. Vous n'avez alors plus à vous occuper de la manière dont les données sont stockées sur le disque dur, de simples instructions permettent d'ajouter, de supprimer, de mettre à jour et surtout de rechercher des données dans une base de données. On peut de plus accéder très facilement à une base de données MySQL à partir de PHP, ce qui permet de développer des sites web très performants et interactifs (par exemple, le forum de Developpez.com). EasyPHP joint PHPMyAdmin à MySQL, un outil écrit en PHP permettant de gérer vos bases de données MySQL. En utilisant EasyPHP, vous pouvez installer un serveur web complet, qui vous permettra de faire tous vos tests de pages PHP en toute facilité. [7]

MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requêtes vers les bases de données relationnelles.

Il en reprend la syntaxe mais n'en a pas toute la puissance.

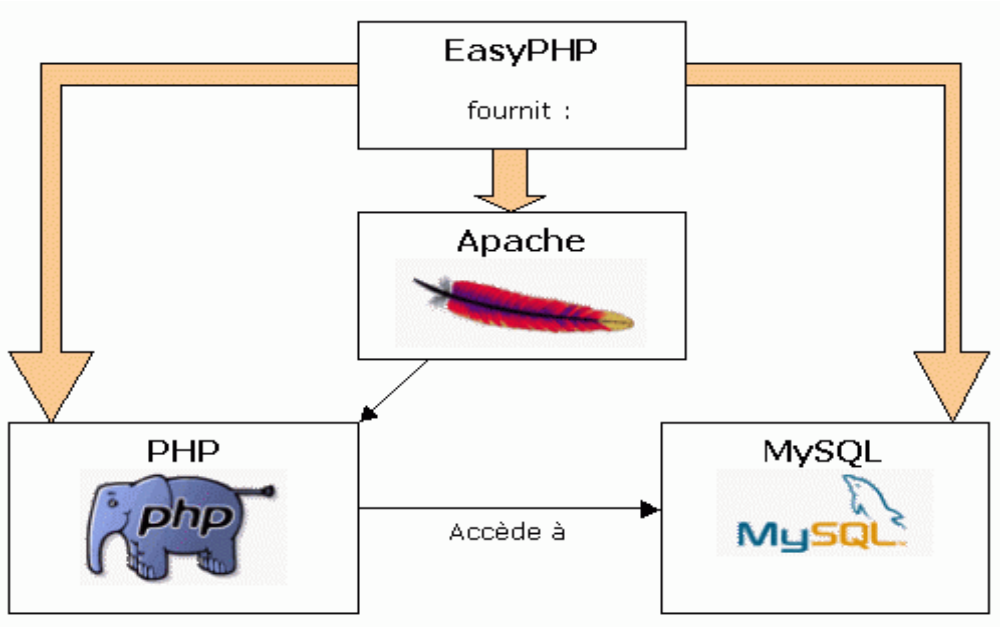

Figure 1 : Relation entre EasyPHP et MySQL. [8]

EasyPHP est un donc paquetage contenant à la fois Apache, PHP et MySQL.

I.3.4.2 Que ce qu'une base de données ?

Une base de données contient plusieurs tables. Une table est un ensemble d'enregistrements. Un enregistrement contient un nombre déterminé de champs (par exemple: login, nom, prénom et date de naissance).
Les champs sont formatés (date, chaîne de caractère, nombre…).
Pas de possibilité de listes chainées, ou d'ensembles. Si une table définit un champ comme clé primaire, chaque enregistrement doit alors différer par sa clé primaire.

I.3.4.3 L'utilité d'une Base de données :

L'utilité d'une base de données n'est plus à prouver et fait partie intégrante des évolutions majeures de l'informatique. Il s'agit de classer, ordonner tous types de données afin d'en faciliter le traitement ultérieur. Une **base** est un ensemble de données organisées en **plusieurs tables** contenant elles même plusieurs **champs**. L'intérêt est de pouvoir **croiser** les données contenues dans ces tables afin d'obtenir un ensemble cohérent.

I.4 L'architecture Client/serveur :

I.4.1Présentation :

De nombreuses applications fonctionnent selon un environnement client/serveur, cela signifie que des **machines clientes** (des machines faisant partie du réseau) contactent un **serveur**, une machine généralement très puissante en termes de capacités d'entrée-sortie, qui leur fournit des **services**. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, etc.

Les services sont exploités par des programmes, appelés **programmes clients**, s'exécutant sur les machines clientes.
Lorsque l'on désigne un programme tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès d'un serveur.
Dans l'architecture client-serveur, une application est constituée de trois parties :
L'interface utilisateur, la logique des traitements et la gestion des données.
•Le client n'exécute que l'interface utilisateur (interfaces graphiques).
•La logique des traitements (formuler la requête).
•Laissant au serveur de bases de données la gestion complète des manipulations de données.

I.4.2 Avantages de l'architecture client/serveur :

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- **Des ressources centralisées** : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.
- **Une meilleure sécurité** : car le nombre de points d'entrée permettant l'accès aux données est moins important.
- **Une administration au niveau serveur** : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.
- **Un réseau évolutif** : grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure.

I.4.3 Inconvénients de l'architecture client/serveur :

L'architecture Client/serveur a tout de même quelques lacunes parmi lesquelles :

- Un coût élevé dû à la technicité du serveur.
- Un maillon faible : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui.

I.4.4 Fonctionnement d'un système client/serveur :

Un système client/serveur fonctionne selon le schéma suivant :

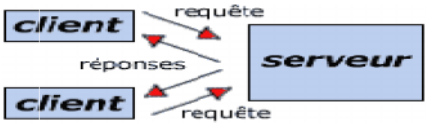

Figure 2 : Fonctionnement Client/serveur. [9]

- Le client émet une requête vers le serveur grâce à son adresse IP (Internet Protocol) et le port, qui désigne un service particulier du serveur.
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine cliente et son port.

I.5 RMI (Remote method Invocation):

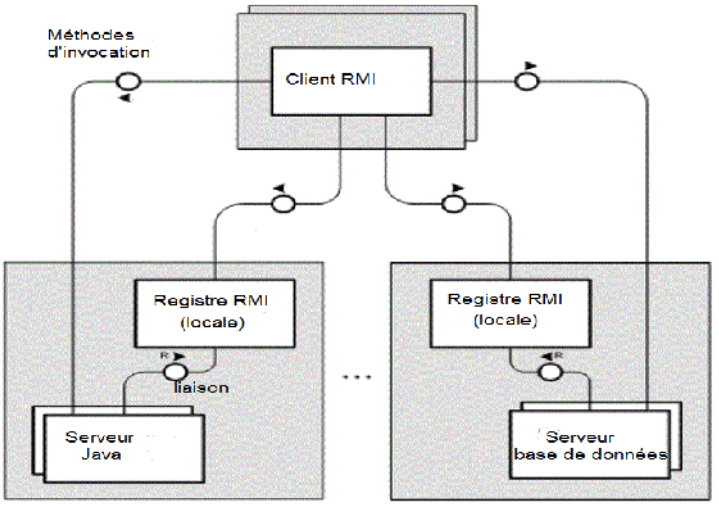

Figure 3 : Fonctionnement de RMI. [10]

I.5.1 Présentation :

RMI est un système d'objets distribués constitué uniquement d'objets JAVA et c'est une API (Application Programming Interface) intégrée à JAVA depuis la version 1.1.
Ce mécanisme permettant l'appel de méthodes entre des objets cet appel peut se faire sur la même machine ou bien sur des machines connectées en réseau.
Les échanges respectent un protocole propriétaire (Remote Method Protocol).

L'idée est de rendre **transparente** la manipulation d'objets distants, un appel de méthode sur un objet distant doit être **syntaxiquement** le même qu'un appel de méthode sur un objet local l'idée aussi est masquer (à l'utilisateur local) les communications nécessaires pour accéder à un objet, en fournissant un objet local de substitution.

I.5.2 L'utilité de RMI :

Le but de RMI est de permettre l'appel, l'exécution et le renvoi du résultat d'une méthode exécutée dans une machine virtuelle différente de celle de l'objet l'appelant. Cette machine virtuelle peut être sur une machine différente pourvu qu'elle soit accessible par le réseau. La machine sur laquelle s'exécute la méthode distante est appelée serveur.

L'appel coté client d'une telle méthode est un peu plus compliqué que l'appel d'une méthode d'un objet local mais il reste simple. Il consiste à obtenir une référence sur l'objet distant puis à simplement appeler la méthode à partir de cette référence.

La technologie RMI se charge de rendre transparente la localisation de l'objet distant, son appel et le renvoi du résultat.

I.6 Les différentes étapes pour créer un objet distant et l'appeler avec RMI :

Le développement coté serveur se compose de :

- La définition d'une interface qui contient les méthodes qui peuvent être appelées à distance.
- L'écriture d'une classe qui implémente cette interface.
- L'écriture d'une classe qui instanciera l'objet et l'enregistrera en lui affectant un nom dans le registre de noms RMI (RMI Registry).

Le développement côté client se compose de :

- L'obtention d'une référence sur l'objet distant à partir de son nom.
- L'appel à la méthode à partir de cette référence.

I.6.1 Le développement coté serveur :

Côté serveur, l'objet distant est décrit par une interface. Une instance de l'objet doit être créée et enregistrée dans le registre RMI.

I.6.2 La définition d'une interface qui contient les méthodes de l'objet distant :

L'interface à définir doit hériter de l'interface java.rmi.Remote. Cette interface ne contient aucune méthode mais indique simplement que l'interface peut être appelée à distance. L'interface doit contenir toutes les méthodes qui seront susceptibles d'être appelées à distance.

La communication entre le client et le serveur lors de l'invocation de la méthode distante peut échouer pour diverses raisons telles qu'un crash du serveur, une rupture de la liaison, etc.

Ainsi chaque méthode appelée à distance doit déclarer qu'elle est en mesure de lever l'exception java.rmi.RemoteException.

I.6.3 Mise en œuvre d'une application avec RMI :

- Définir une interface distante (Xyy.java).
- Créer une classe implémentant cette interface (XyyImpl.java).
- Compiler cette classe (javac XyyImpl.java).
- Créer une application serveur (XyyServer.java).
- Compiler l'application serveur.
- Démarrage du registre avec rmiregistry.
- Lancer le serveur pour la création d'objets et leur enregistrement dans rmiregistry.
- Créer une classe cliente qui appelle des méthodes distantes de l'objet distribué (XyyClient.java).
- Compiler cette classe et la lancer.

Conclusion

Nous avons parlé dans ce chapitre de la gestion d'une banque et la sécurité bancaire comme nous avons présenté les outils utilisés dans notre PFE comme Java et la mise en place de ce programme a nécessité un environnement de développement intégré comme NetBeans, aussi qu'un gestionnaire de base de données MySQL sous le paquetage EasyPHP pour la création de la base de données , suivi de l'architecture Client/serveur aussi le mécanisme utilisé RMI.

Dans le chapitre suivant, nous présenterons en détails les différents résultats qu'on a réalisé et quelques explications de notre application.

Chapitre II: Développement d'une application de gestion d'une banque avec Java RMI.

II.1 Introduction :

Dans ce chapitre nous allons présenter la partie application de notre projet de fin d'études qui consiste à développer une application de gestion d'une banque avec Java RMI, pour cela nous allons détailler les différentes étapes à suivre pour la réalisation de notre projet dans ce sui suit.

II.2 Modélisation avec UML:

Dans ce qui suit et avant de présenter nos tables de données et les différentes interfaces, nous allons présenter quelques diagrammes de classes, de séquences et des cas d'utilisations de notre application:

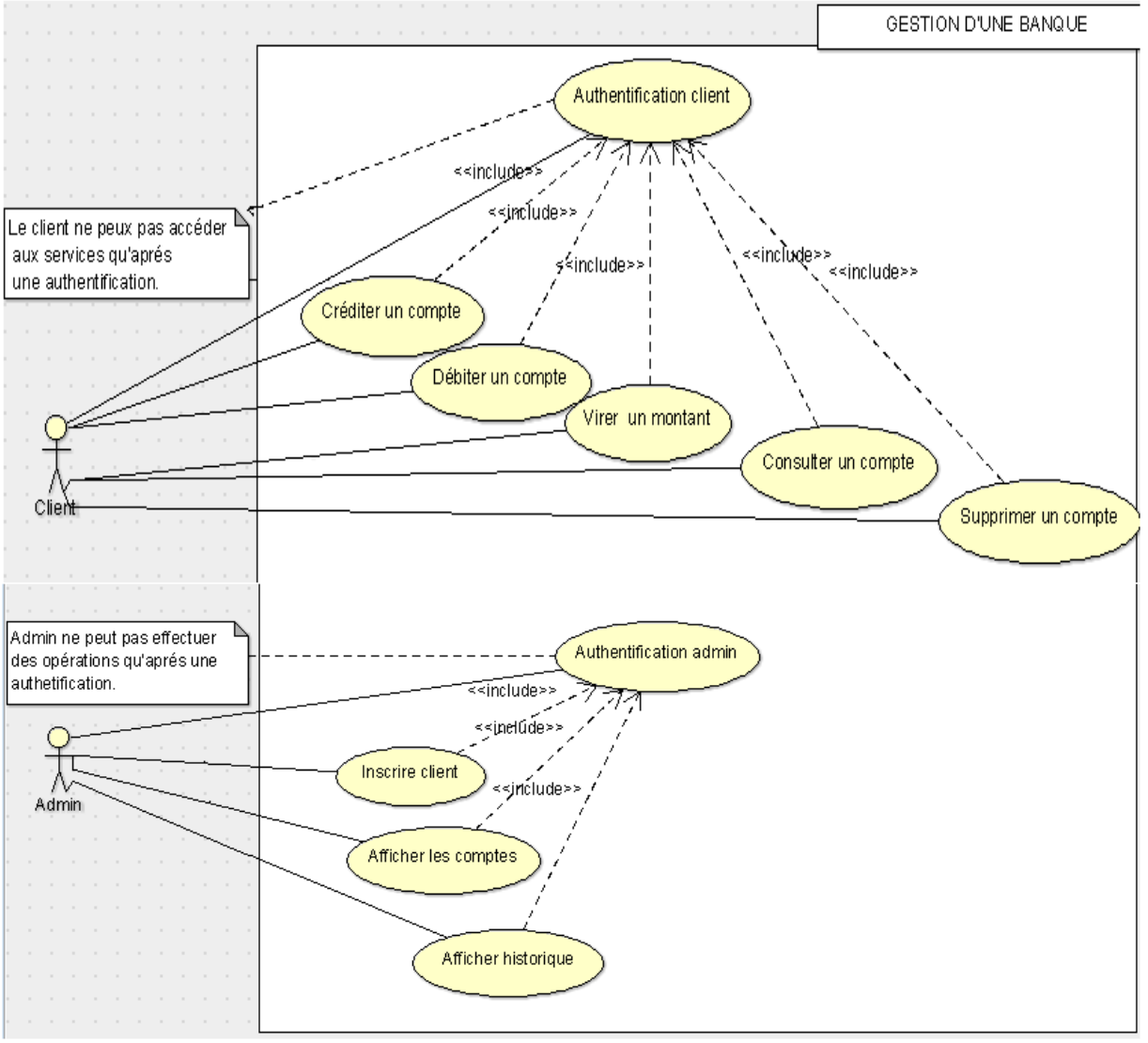

Figure 4: Diagramme de cas d'utilisation.

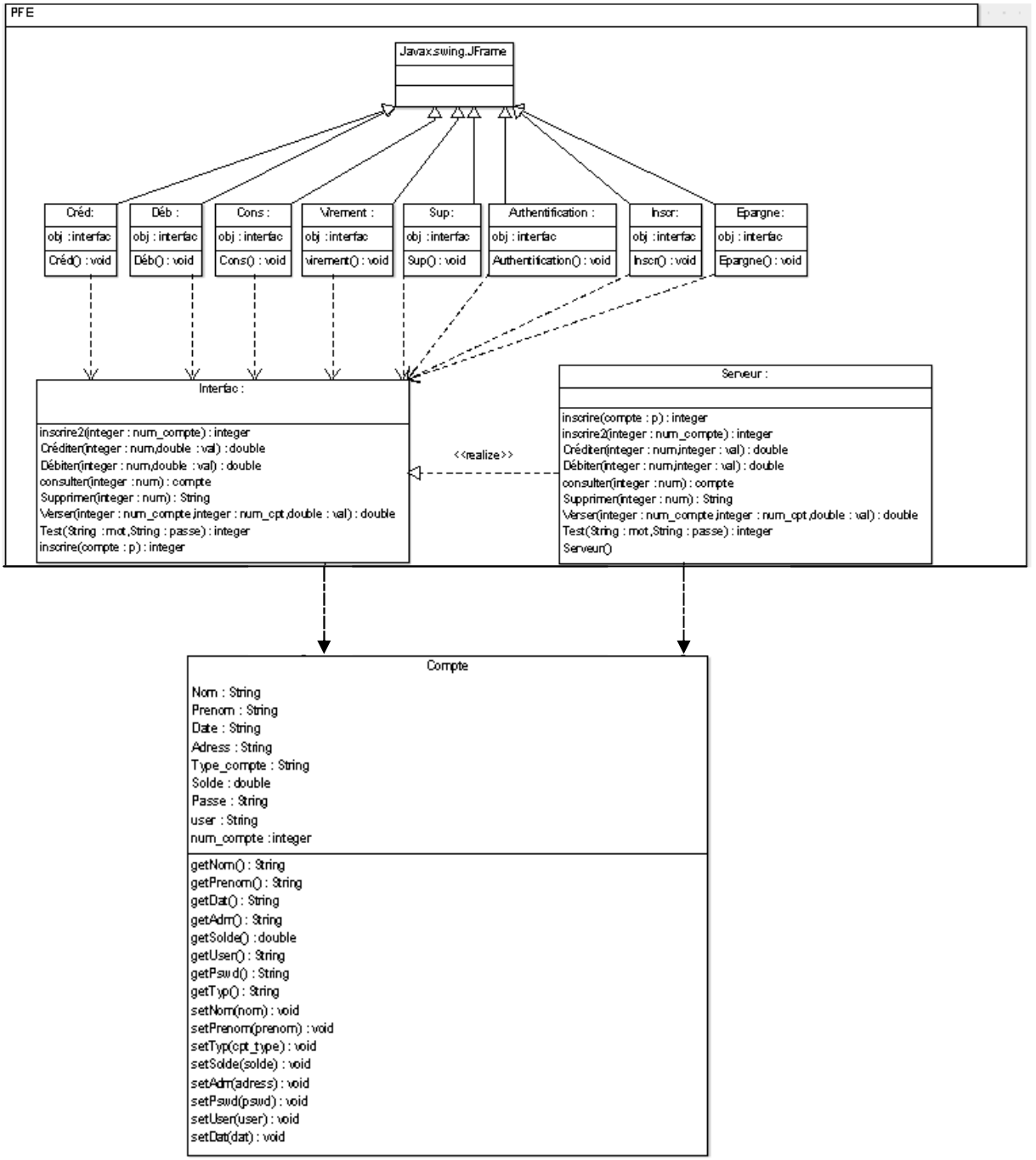

Figure 5: Diagramme de classe.

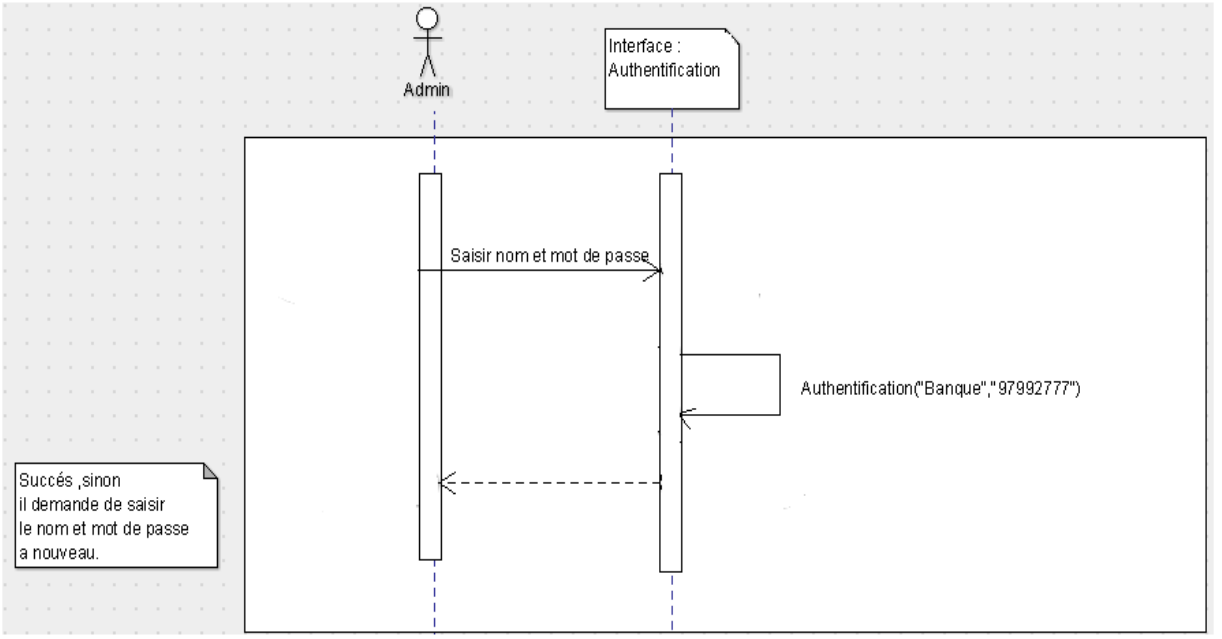

Figure 6: Diagramme de Séquence: Authentification administrateur.

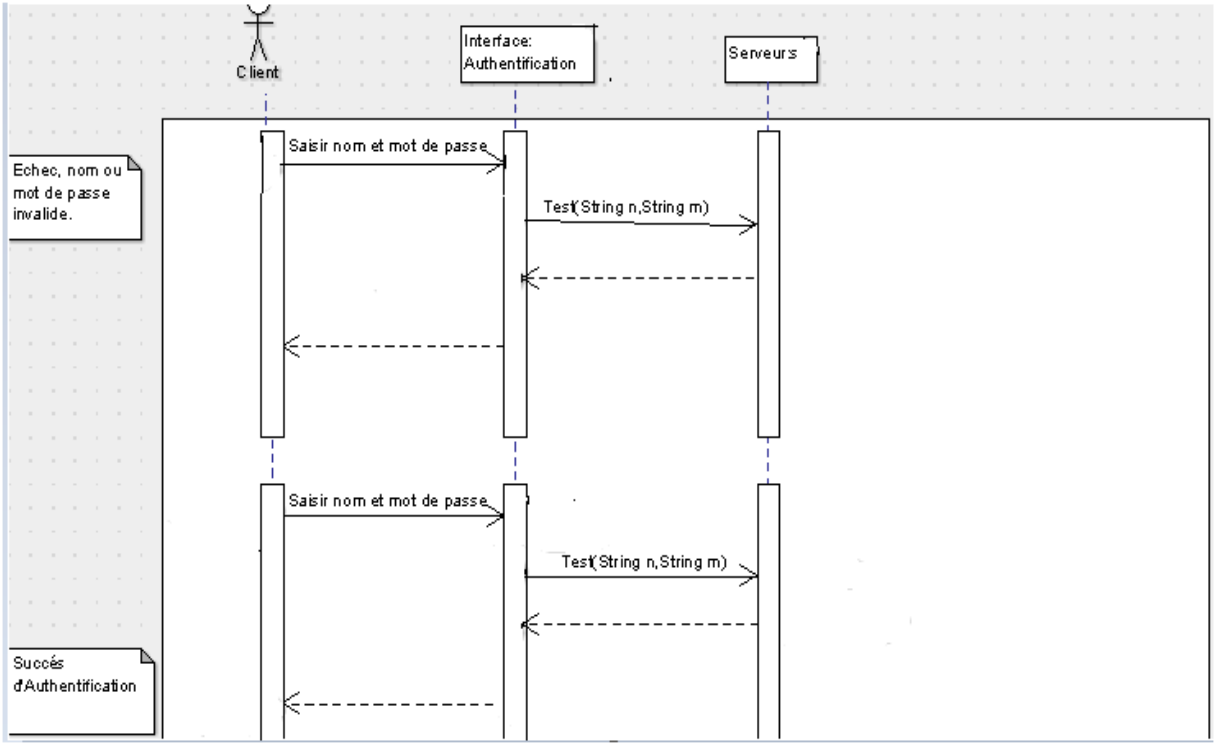

Figure 7: Diagramme de séquence: Authentification client.

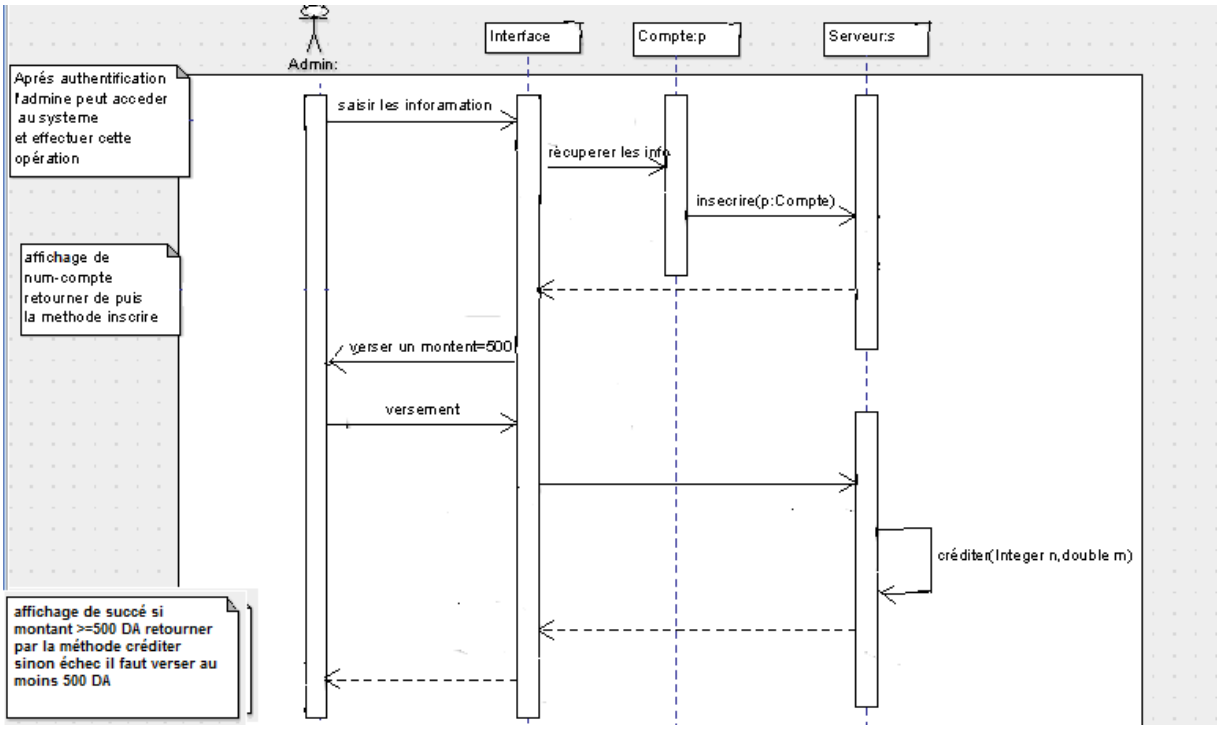

Figure 8: Diagramme de séquence: inscrire.

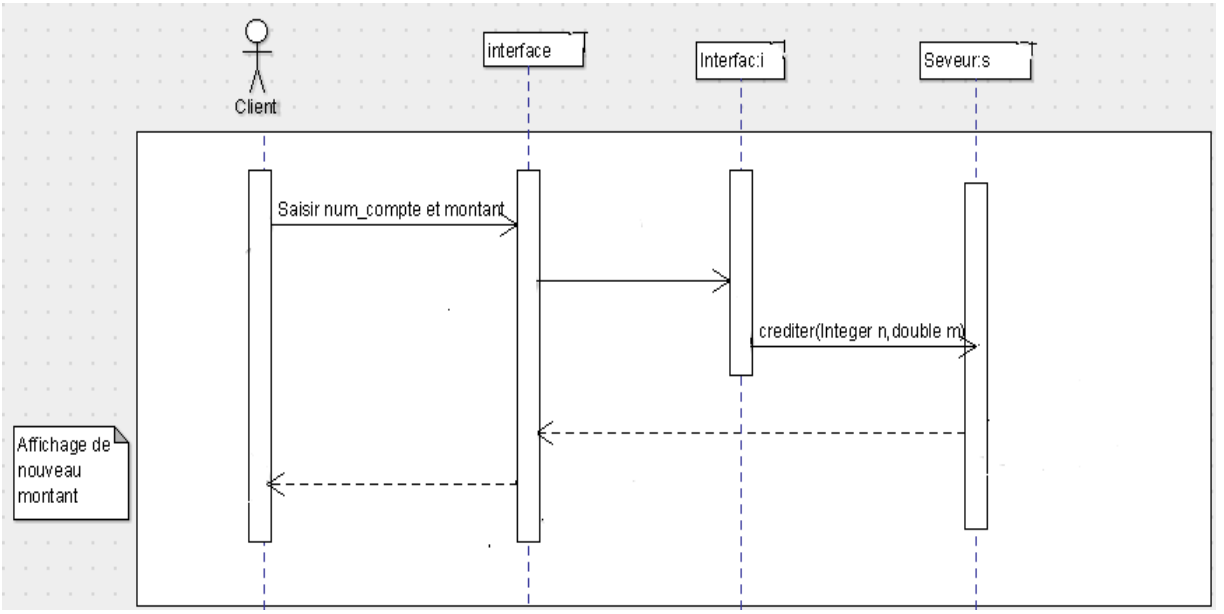

Figure 9: Diagramme de séquence: Créditer.

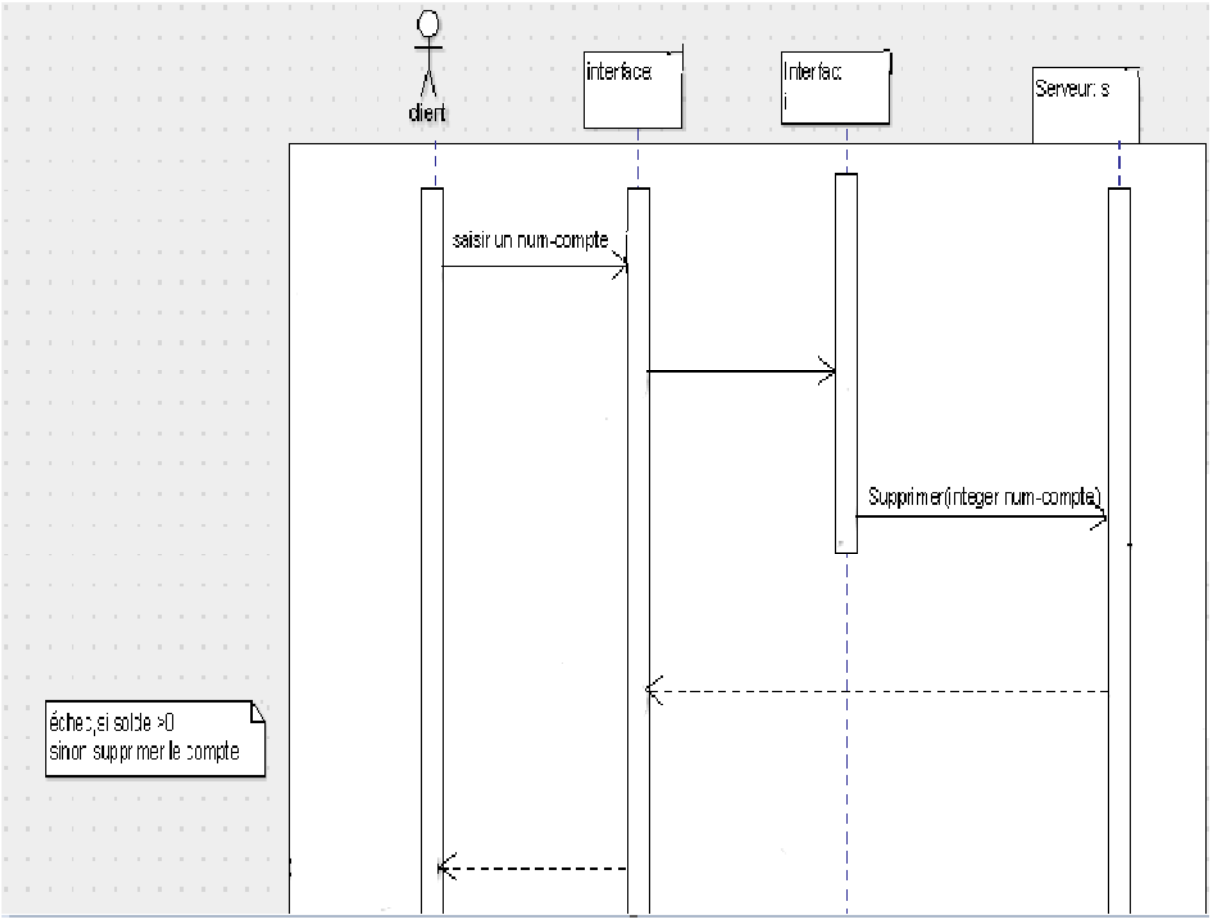

Figure 10: Diagramme de séquence supprimer.

II.2 Tables de notre base de donnée :

Nous avons commencé par la création d'une base de données qu'on l'a nommée « PFE » ensuite dans cette base nous avons créé deux tables: « historique » et « compte », comme indiqué ci-dessous:

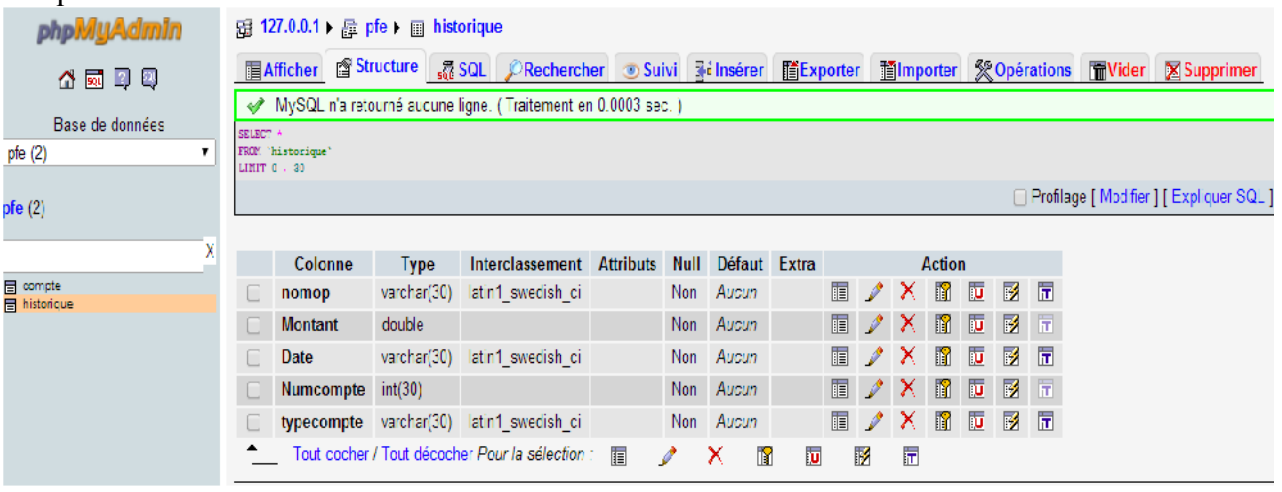

Figure11: La table historique.

La table historique comprend cinq champs: le type de l'opération faite par le client, le montant géré, la date de cette opération, le numéro de compte du client qui a fait l'opération et

enfin le type de compte.

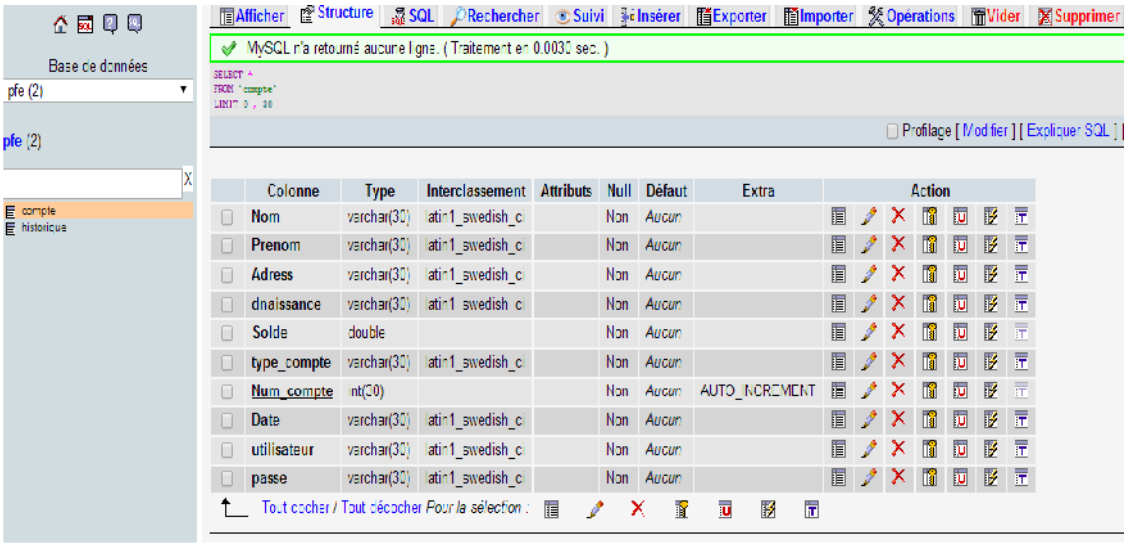

Figure 12: La table Compte.

La table « Compte » contient des champs indiquant des informations sur les clients.

II.3 Lancement du projet :

Figure 13: L'authentification.

La sécurité est nécessaire pour n'importe quelle application, c'est pour cette raison que nous avons crée un mécanisme d'authentification pour l'ensemble des clients mais aussi pour l'administrateur. Donc, il y a un login et mot de passe pour chaque client et aussi un login et mot de passe pour l'administrateur. Les login(s) et mots de passe sont donnés aux clients après leurs inscriptions par l'administrateur.

Un mauvais mot de passe ou login est aussi géré par notre application en cas d'erreurs de saisie comme indiqué dans la figure suivante :

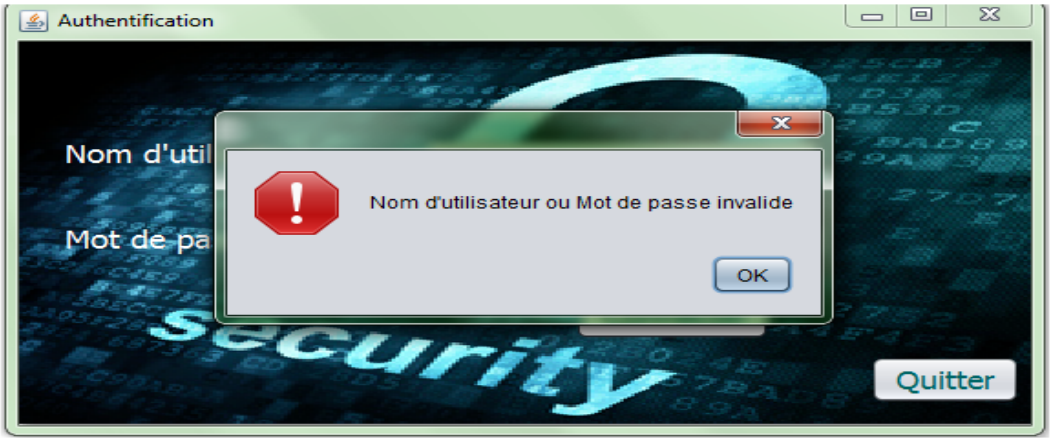

Figure 14: Accès interdit (mauvais mot de passe).

II.3.1 Partie administrateur :

L'IHM suivante est celle de l'administrateur après son authentification.

Figure 15 : Partie administration.

II.3.1.A Inscription : L'administrateur peut inscrire des clients (ouvrir des comptes courants) et ouvrir des comptes épargnes pour les clients qui sont déjà inscrits.

Figure 16 : Inscription (ouvrir des comptes).

II.3.1.A.1 Ouvrir un compte courant :

L'administrateur est chargé pour ouvrir des comptes courants (inscrire des clients pour la première fois), et l'inscription est comme suite :

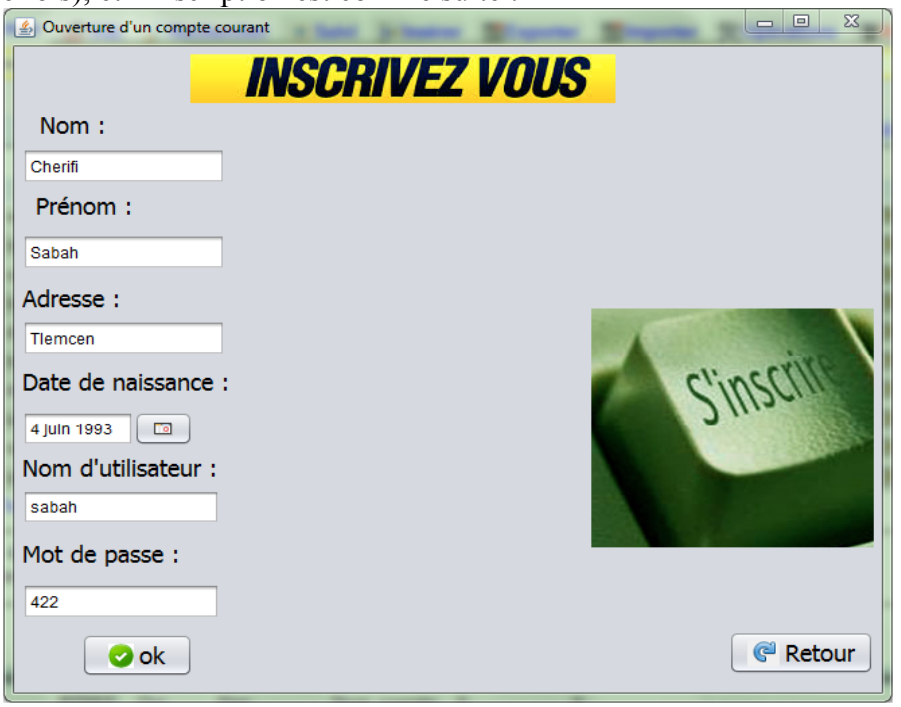

Figure 17: Ouvrir un compte courant.

Pour ouvrir un compte courant il faut verser au moins 500DA, les figures suivantes montrent les résultats du versement initial de l'inscription :

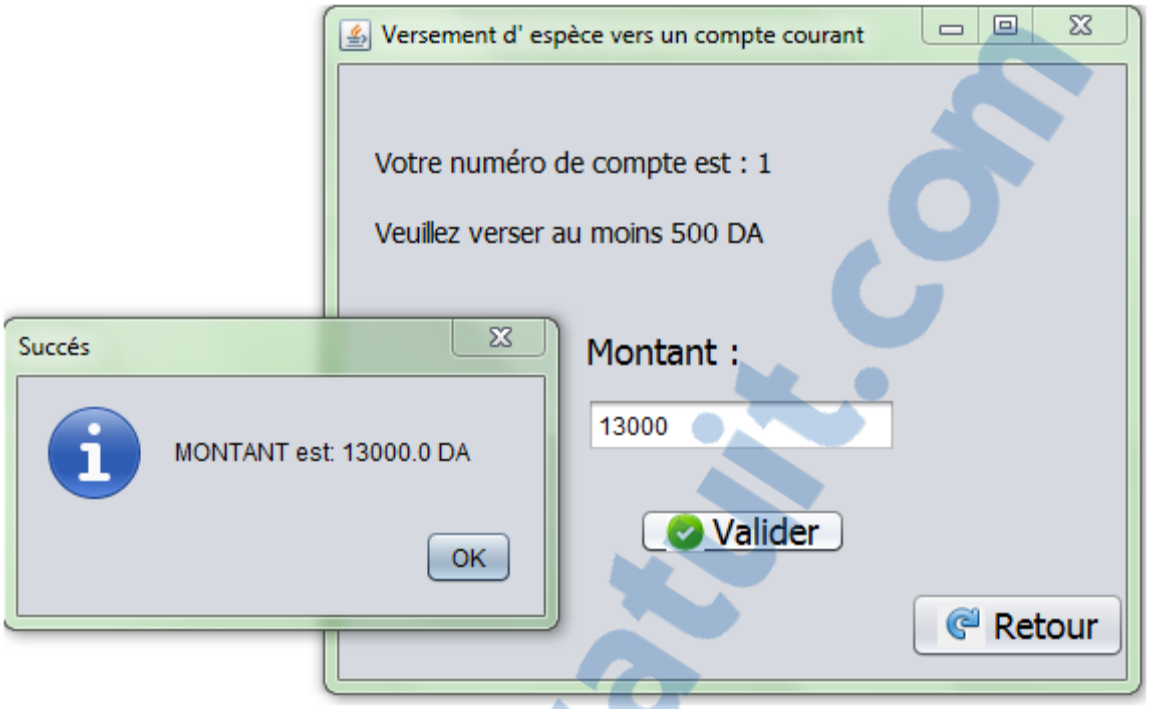

Figure 18: Versement vers le compte courant.

II.4.1.A.2. Ouvrir un compte épargne :

Un client inscrit a la possibilité d'ouvrir un compte épargne et il est obligé de verser au moins 10000DA, cette opération nécessite d'entrer le numéro du compte courant et de verser le montant nécessaire.

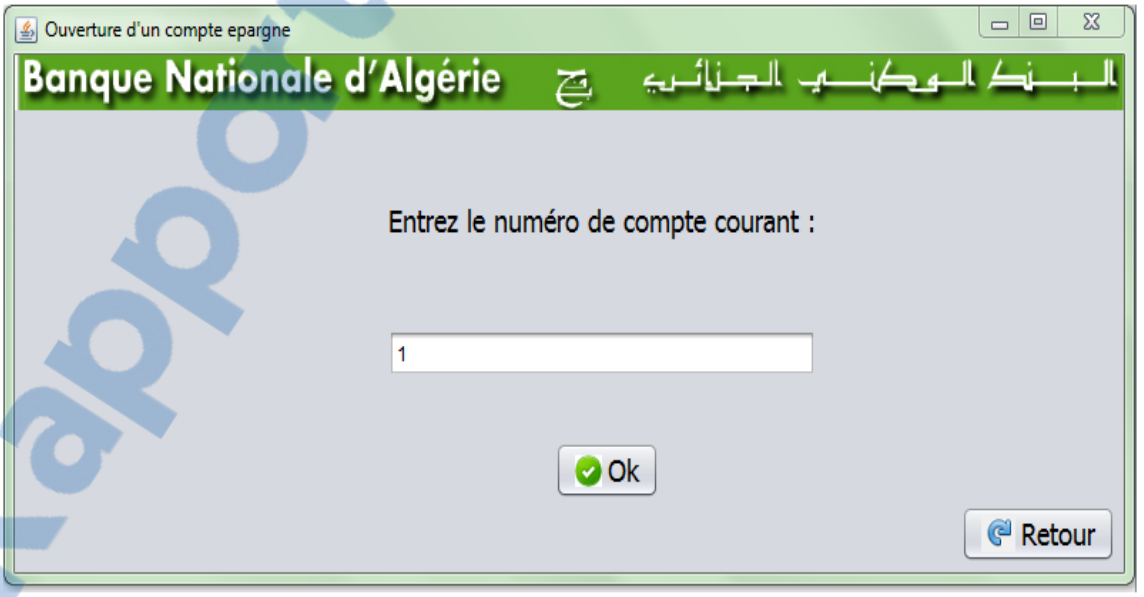

Figure 19: Ouvrir un compte épargne.

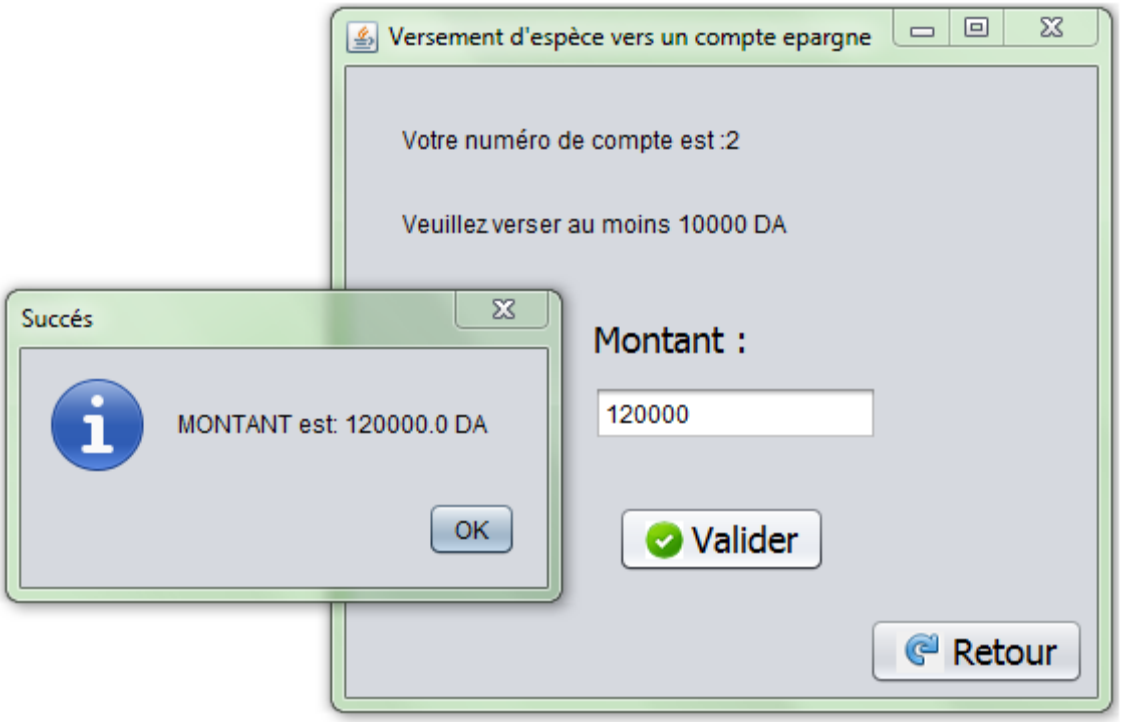

Figure 20: Versement du montant.

Un administrateur peut aussi faire la gestion des clients en cas où il y aura un conflit il peut vérifier les comptes comme il peut vérifier l'historique des comptes:

Figure 21: Gestion des clients.

« Les comptes » comprend la liste de tous les comptes courants et épargnes de tous les clients.

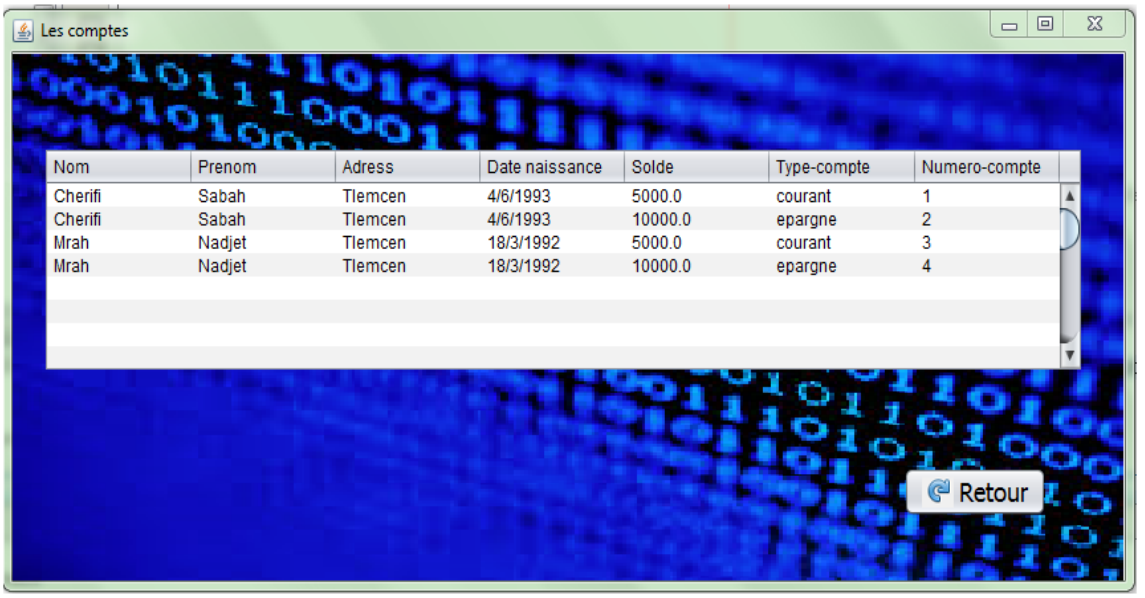

Figure 22: La liste des comptes.

« Historique » donne aux administrateurs la possibilité de voir les historiques des comptes courants et épargnes, il faut que l'administrateur entre le numéro de compte du client voulu et valider sa demande:

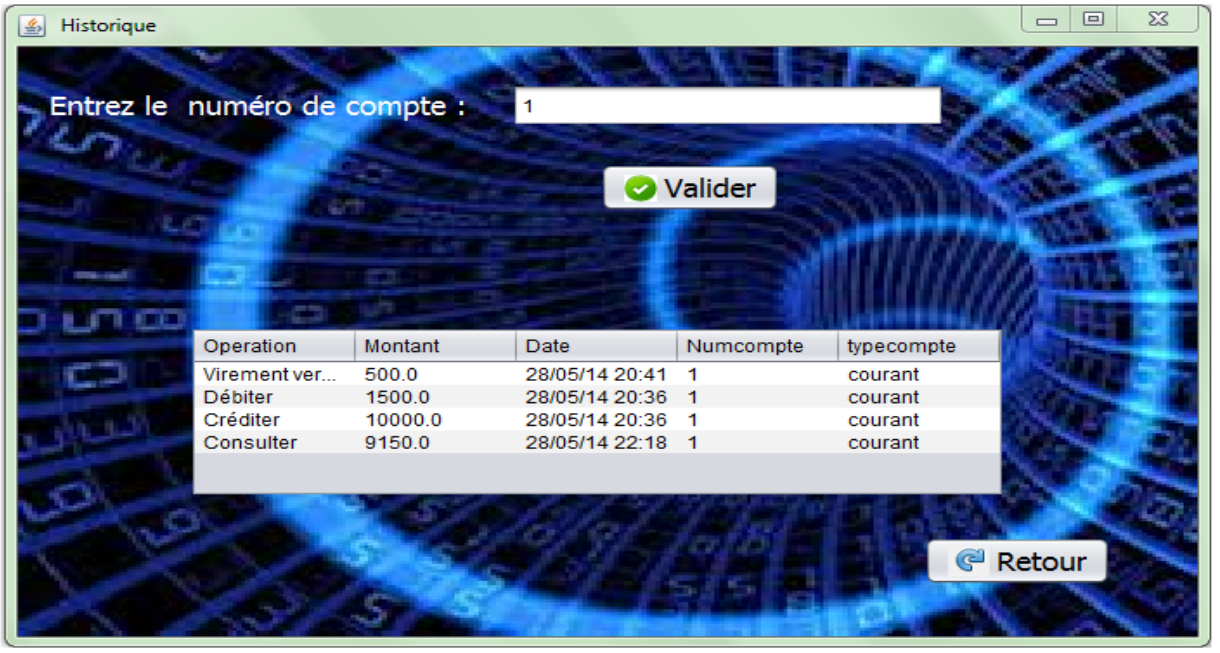

Figure 23: Historique d'un compte.

II.4.2 Partie client :

Un client inscrit à la banque peut accéder à un ensemble de services sans passer par un administrateur, L'IHM suivante est celle des clients:

Figure 24: Partie client.

Cette partie donne aux clients la possibilité de gérer leurs comptes et faire quelques opérations: **Créditer, Débiter, Virement vers un autre compte, Consulter et Fermer le compte** s'il est vide. Donc nous allons présenter dans ce qui suit l'ensemble des IHMs proposées aux clients en fonction de leurs demandes.

Figure 25: La liste des opérations bancaires.

II.4.2.1 Créditer

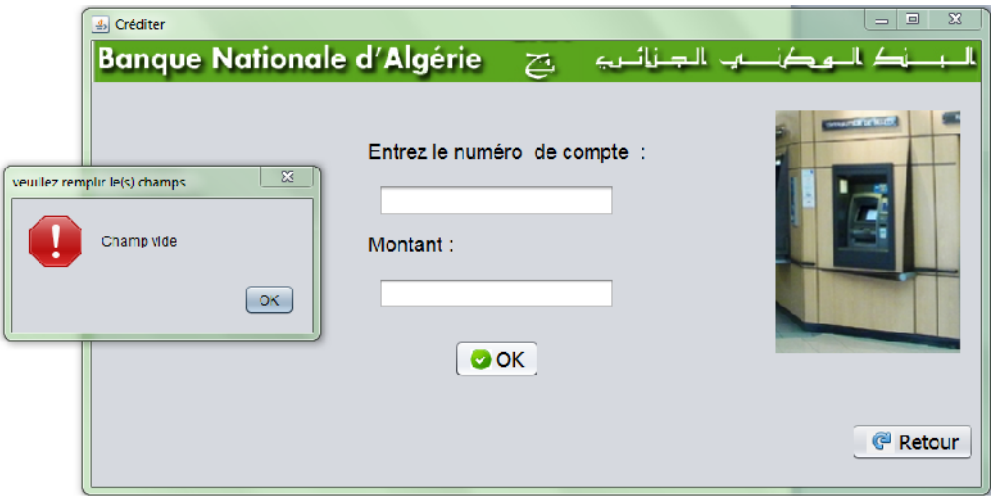

Figure 26: Cas où le(s) champ(s) reste vide(s).

Il est impossible de valider l'opération **Créditer** si les champs sont vides.

Il est aussi impossible de valider l'opération si on ne respecte pas le type des champs.

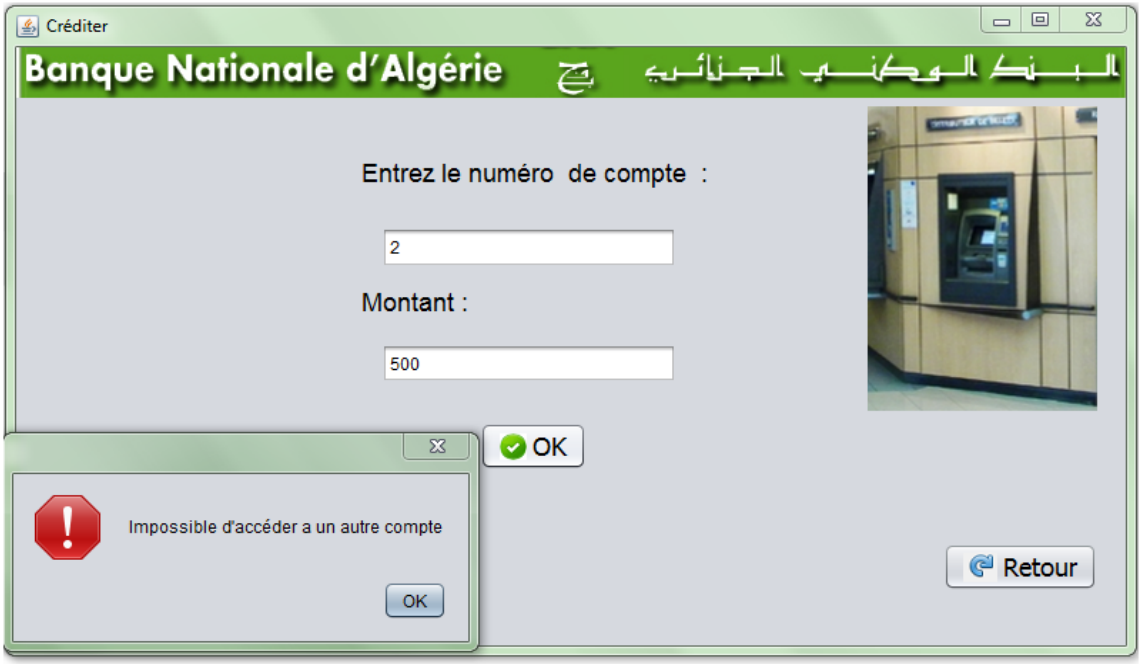

Figure 27: La sécurité des comptes.

Un client ne peut gérer que son propre compte, il n'a pas le droit d'accéder aux autre comptes comme indiqué dans la figure précédente.

Figure 28: Succès de l'opération créditer.

II.4.2.2 Débiter

Les mêmes remarques précédentes sont valables pour l'opération Débiter.

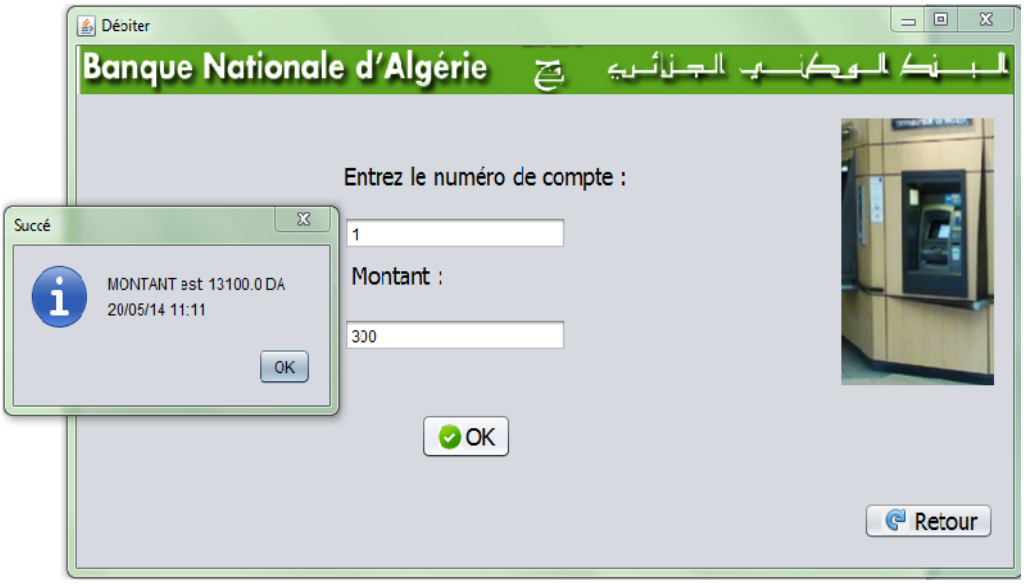

Figure 29: Succès de l'opération Débiter.

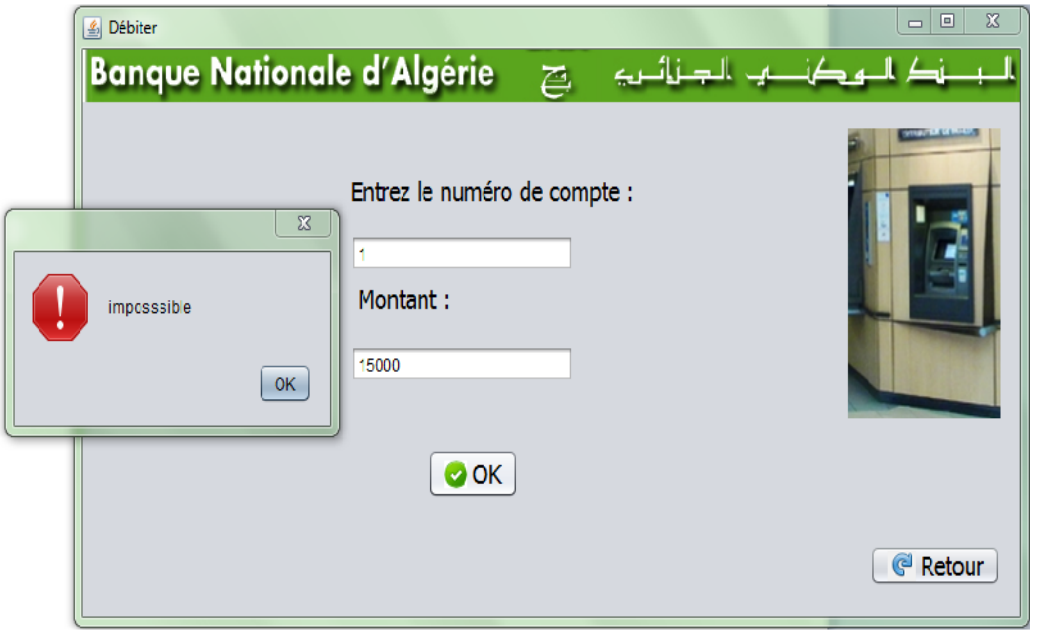

Figure 30: Opération impossible.

La figure précédente montre qu'on ne peut pas débiter un compte sauf si son montant le permet.

II.4.2.3 Virement

Cette opération demande de saisir le numéro de compte de l'émetteur et le numéro de compte du récepteur et le montant du virement.

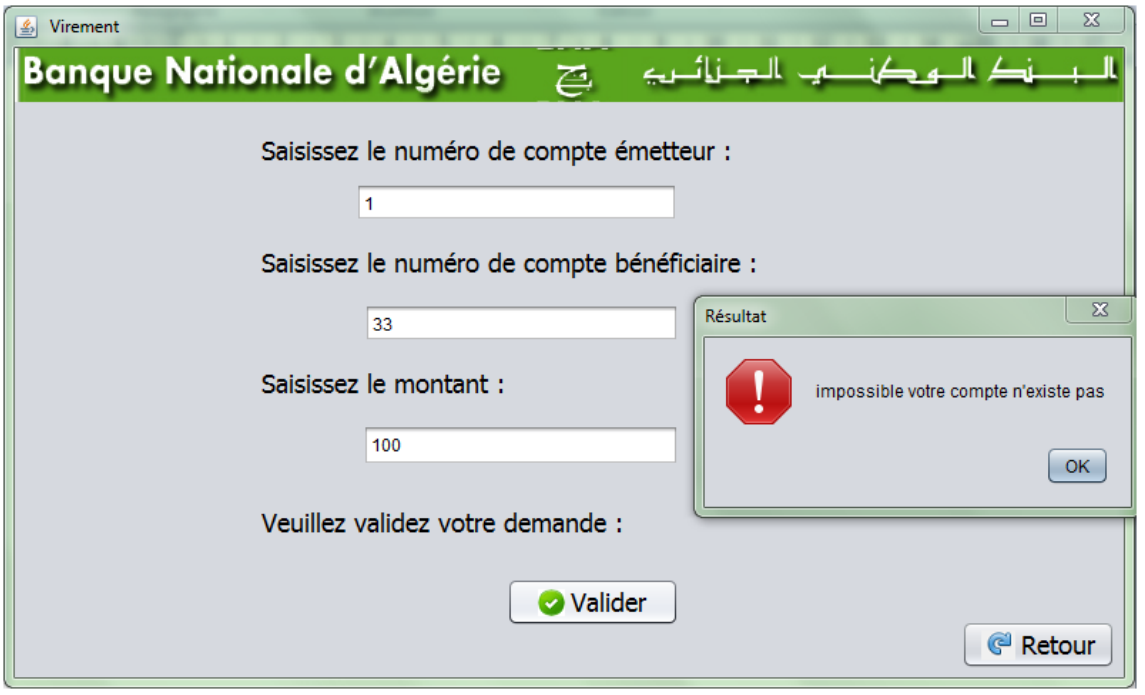

Figure 31: Compte destinataire n'existe pas.

Il est impossible de faire un virement vers un compte qui n'existe pas.

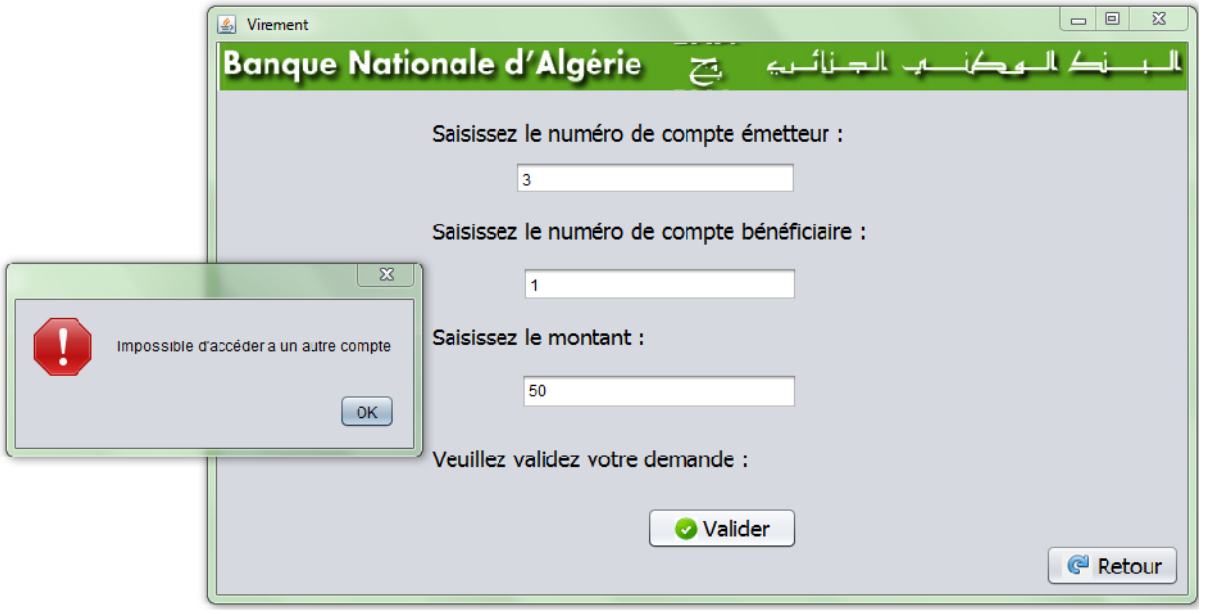

Figure 32: Compte destinateur invalide.

Il est impossible de faire le virement d'un compte d'un autre client.

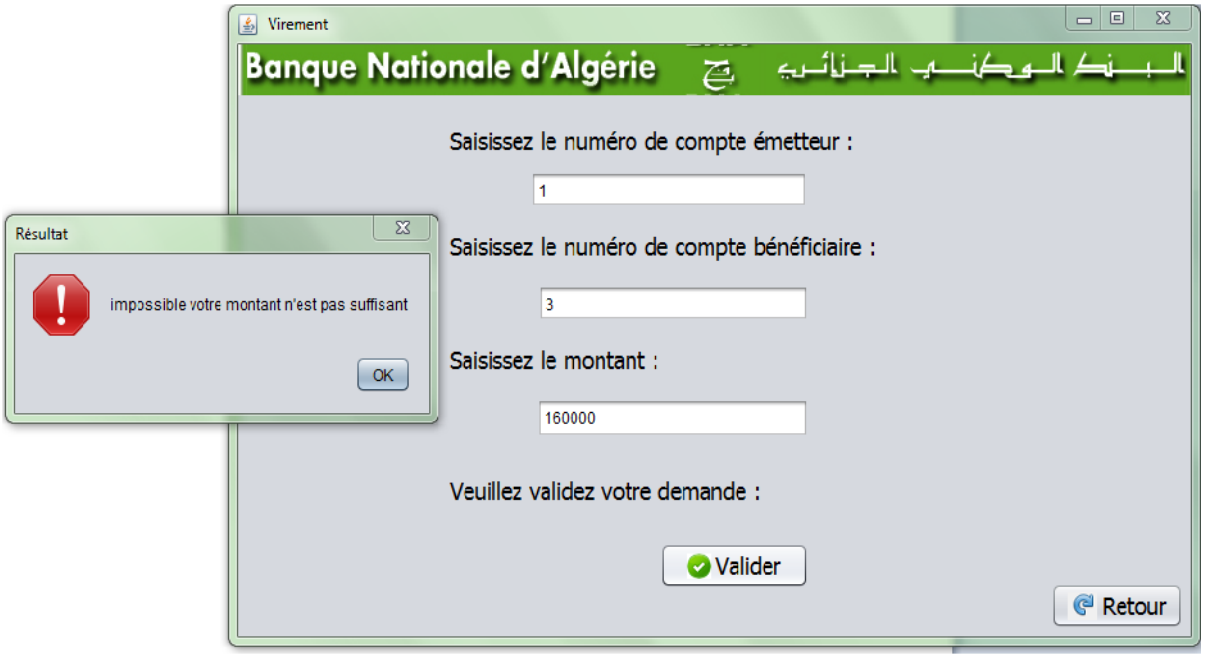

Figure 33: Montant insuffisant.

Il est aussi impossible de faire le virement si le compte émetteur ne contient pas le montant indiqué.

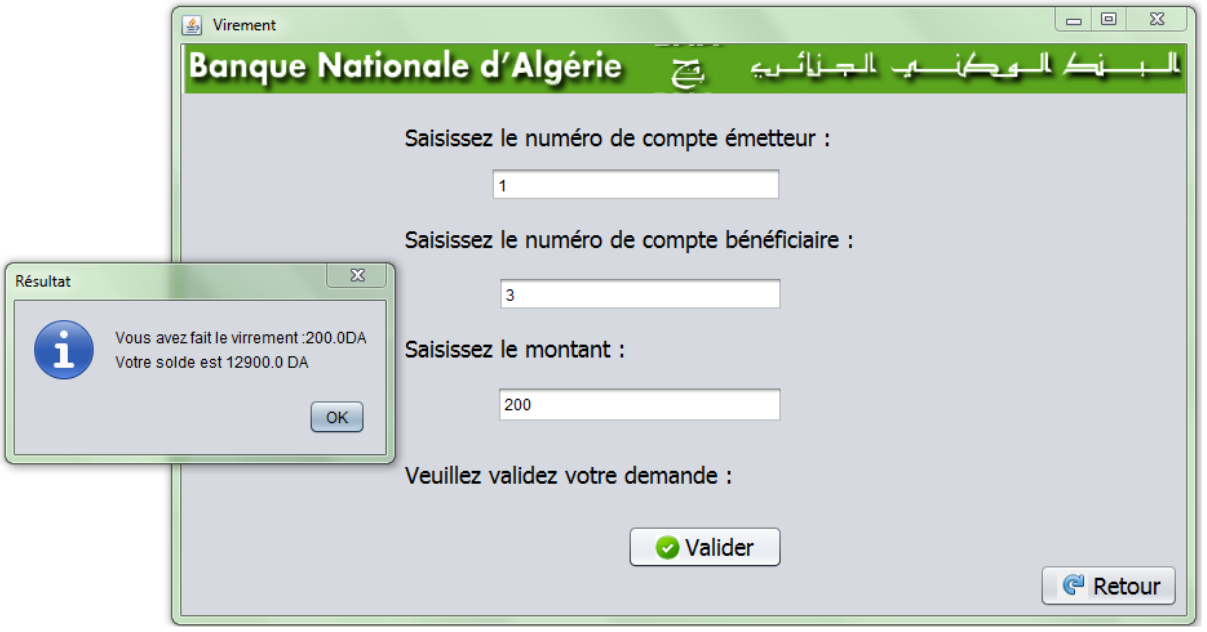

Figure 34: Succès de l'opération Virement.

II.4.2.4 Consultation

La consultation d'un compte est comme suit :

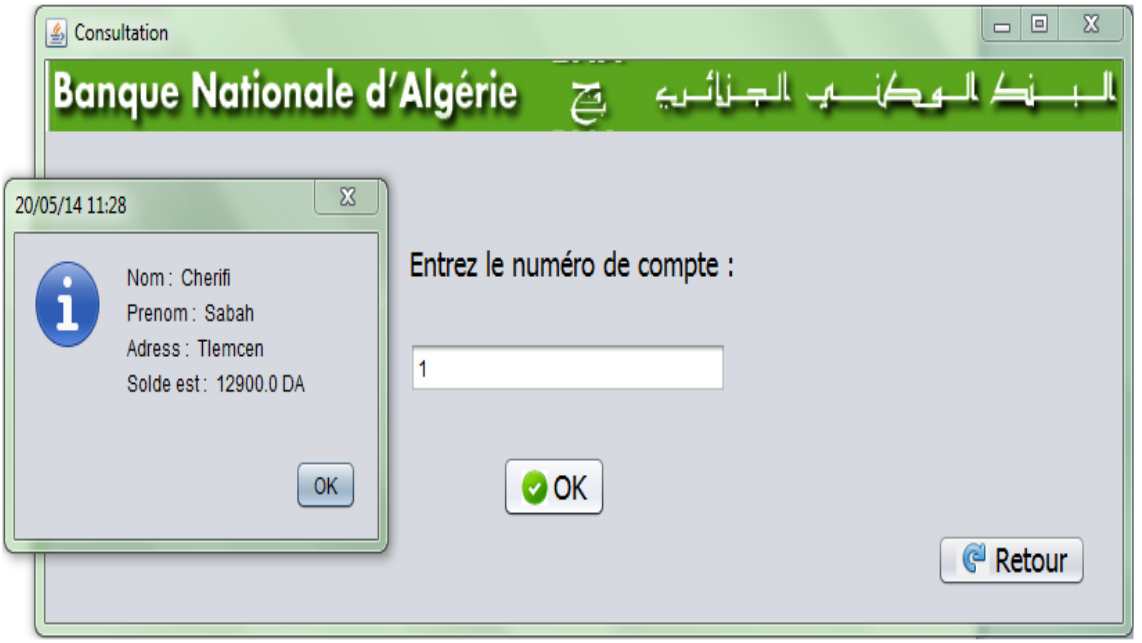

Figure 35: Consultation d'un compte.

II.4.2.5 Supprimer :

Cette opération de suppression d'un compte ne demande qu'à entrer le numéro de compte.

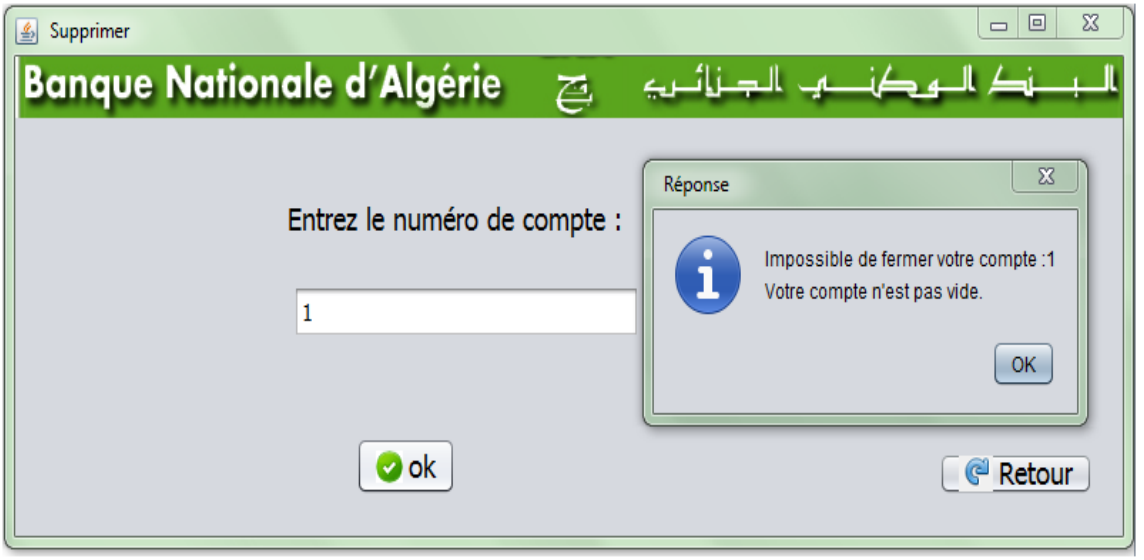

Figure 36: Opération Supprimer échouée.

Pour supprimer un compte, il est primordial de le vider.

Figure 37: Suppression d'un compte vide.

Enfin et en cas où le client trouve une difficulté ou un problème, il a la possibilité de rejoindre les responsables pour régler ses problèmes comme indiqué dans les figures suivantes.

II.4.2.6 : Aide proposé aux clients :

Figure 38: Client a besoin d'aide.

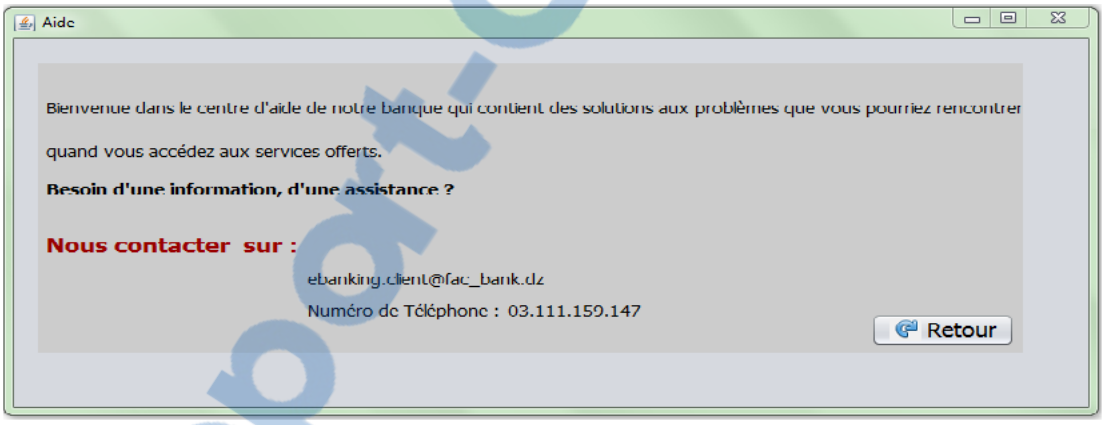

Figure 39: Aide aux clients.

Et enfin la partie client contient aussi des informations sur notre PFE le thème, les réalisateurs et les encadreurs du projet :

II.4.2.7: Présentation de notre Projet de Fin d'Études

Figure 40: À propos notre PFE.

Conclusion:

Ce chapitre présente tout les résultats de notre projet et montre tout les cas et les opérations possibles et impossibles lors de la gestion d'une banque et aussi explique la notion de la sécurité bancaire.

Conclusion générale :

Dans ce PFE, nous avons conçu et réalisé une application Client/serveur de gestion d'une banque avec Java RMI.

Cette application offre une sécurité totale des comptes bancaires et respecte tout les services d'une banque à un client et ses droits comme les devoirs d'un administrateur.

Nous avons défini et évoqué les outils utilisées pour réaliser notre application, ainsi que le déroulement de l'application et les résultats obtenus.

D'autre part, ce PFE nous a aidés à mieux comprendre le langage de programmation« JAVA» et surtout le mécanisme RMI et en même temps d'avoir une idée globale sur le système de base de données et ses fonctionnalités, alors nous trouvons que ce PFE a ajouté et a développé nos connaissances et nos idées.

Résumé :

Le travail réalisé dans le cadre de ce PFE consiste à développer une application de gestion d'une banque avec Java RMI. Notre rapport est constitué de deux chapitres. Le premier chapitre est consacré à la présentation des outils utilisés comme JAVA en particulier RMI et SWING, mais aussi NetBeans et MySQL. Nous avons également présenté dans ce chapitre un aperçu global sur les architectures de type client/serveur. Le deuxième chapitre présente l'application réalisée dans le cadre de ce PFE.

MOTS CLES : **Java, RMI**, **NetBeans, EayPHP et MySQL.**

Summary:

The objective of our work is to develop an application to manage a bank with Java RMI. Our report consists of two chapters. The first chapter is devoted to the presentation of the tools used in particular JAVA, SWING, RMI, NetBeans and MySQL. We also presented in this chapter an overview on client / server architecture. The second chapter presents our application.

KEYWORDS : **Java, RMI, NetBeans, EayPHP and MySQL.**

ملخص :

في مشروعنا قدمنا منهاج زبون\خادم باستخدام جافا RMI.

يتكون مشروعنا من فصلين :

في الفصل الأول قدمنا إدارة البنك كما قدمنا المعدات اللازمة لتحقيق مشروعنا بدء بتعريف اللغة الموجهة المنحى" Java" ثم "NetBeans" بالتركيز على آلية "RMI" وحزمتها "EayPHP".

وبعد ذلك عرفنا نظام إدارة قواعد البيانات العلاقية "MySQL"

في النهاية قدمنا هندسة زبون\خادم.

ويعرض الفصل الثاني نتائج المشروع النهائية.

الكلمات الرئيسية هي : MySQL. و Java, RMI, NetBeans, EayPHP.

Références bibliographiques

[1] http://maurise-software.e-monsite.com/medias/files/java-6-2.pdf

[2] ***Développons en Java*** **Jean-Michel DOUDOUX.**

[3] www.developpez.com **club des développeurs**

[4] Java Swing tutorial Jan Bodnar

[5]http://www-igm.univ-mlv.fr/~dr/XPOSE/Site/Version_redigee/ArgoUML.htm

[6] Oracle Corporation and/or its affiliates. Sponsored by ORACLE

[7] Atef GADHOUMI Serendipity (Moteur de blog) Manuel technico-pédagogique.

[8] http://geronimo.developpez.com/EasyPHP/images/schema.gif

[9]http://static.commentcamarche.net/www.commentcamarche.net/pictures/cs-images-cs.gif

[10] http://twimgs.com/ddj/images/article/2008/0811/081101oh01_f1.gif